Technical Report ITL/ANTD - 93-4

September 1996

Multipoint Communication Service Over

Multicast Networks

Dr. RachidSijelmassi

Brett Strausser

GuillaumeSchaff

Stephane Violet

Multimedia Protocols Project

Multimedia and Digital Video Group

National Institute of Standards and Technology

Rationale for the Research

The Multimedia Protocols Project is focusing attention in the area of multimedia communications. Our interests lie in promoting standardization in this emerging technology and in advancing the state of the art. Toward this end we are working with the International Telecommunications Union recommendations on transmission protocols for multimedia data. These standards deal with transmitting data in various formats (audio, video, data) in an efficient, secure and flexible manner.

One weakness in this set of recommendations is the absence of multicast networks. Recommendation T.122/T.125 (Multipoint Communication Service) addresses the problem of providing data delivery to multiple recipients of an audiographics and/or audiovisual conference. Connections established between participants in a conference are strictly point to point in nature, regardless of any native multicast network infrastructure. The result is duplicative transmissions on point to point connections to participants on a multicast network.

Using point-to-point transport connections to implement multicast is inherently sub-optimal in several situations. Examples include LANs using shared media (e.g., Ethernet LANs), wireless networks, the MBONE, etc. However, there are no provisions in T.123 for using native multicast services when they are available.

Stop-Gap Solution: Convergence Layer

The Multimedia Protocols Project is researching the provision of multicast support by constructing a "Convergence Layer" protocol between MCS and the underlying transport. This new layer will mask the use of multicast network services. Our intention is to test the multicast network protocol for robustness and scalability in a multimedia conferencing scenario. This approach allows the maintenance of interoperability with existing implementations of T.122.

Final Solution: "Multicast Network Aware MCS"

Our intent is to promote research into incorporating "multicast awareness" into the MCS protocol itself, thus eliminating the need for a "Convergence Layer". We hope to make this recommendations to the ITU for the next generation of the MCS standard.

Background

MCS provides the multicast transport capabilities needed by higher layers of the T.120 stack of multimedia protocols. The MCS service and protocol are defined in documents T.122 and T.125 of the ITU's T.120 series, respectively.

MCS provides the following functions:

- domain management
- channel management
- distributed token management
- hierarchical multicast transport.

MCS domains are hierarchical and correspond to communication trees whose nodes are called *Multipoint Communication Units* (MCUs), as illustrated in Figure 1 below.

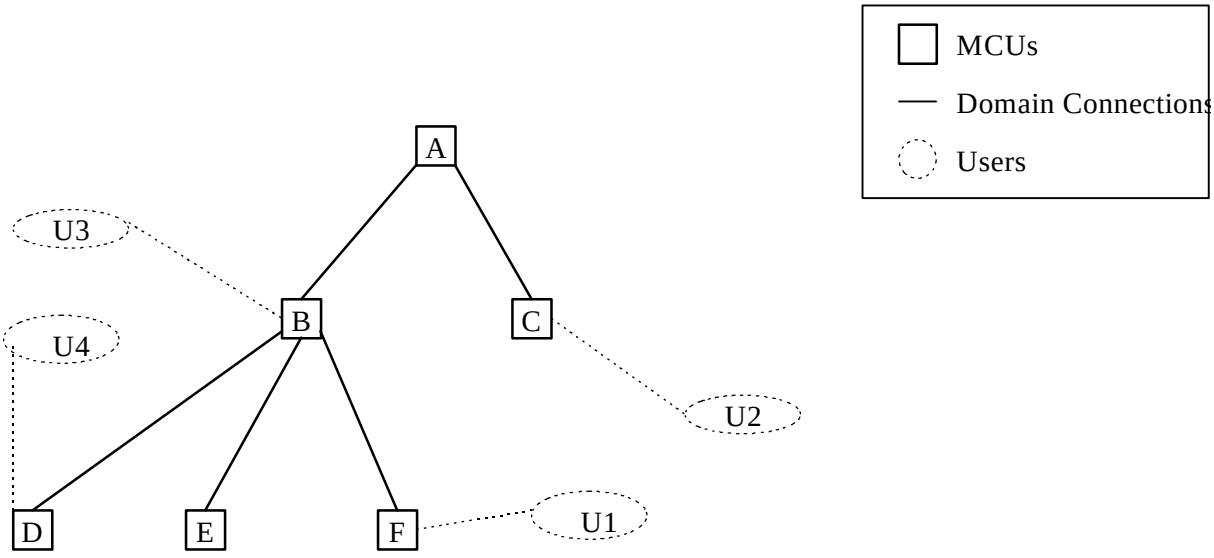

Figure 1: **An MCS Domain – MCUs and attachements**

MCS Users who participate in a conference (or domain) are attached to MCUs. In practice, an MCU and its users can be on a same computer or can be connected through a LAN or other means. The T.120 standards do not specifiy the interface between users and MCUs other than by defining available service primitives.

Domain Connections, or connections among MCUs, consist of one or more point-to-point Transport Connections. Several transport connections are used to implement priorities.

MCS *Domain Management* consists primarily of connection management.

MCS *Channels* are overlapping subsets of a domain. Any data sent on a channel will be multicast to all users who have subscribed to that channel. The MCUs implement hierarchical multicasting using the underlyting poit-to-point transport connections among them.

MCS also provides two types of distributed token management schemes: one to allow for global mutual exclusion, the other to support multiparty rendezvous.

As mentioned above, the MCS protocol (T.125) explicitly relies on point-to-point transport stacks. The T.120 series of recommendations includes recommendation T.123 which defines profiles offering point-to-point, reliable, transport service over several technologies (OSI,ISDN,TCP/IP...).

Convergence Layer Architecture

The requirements guiding our design are as follows:

- allow MCS to use native multicast for data transfer, in LANs and WANs
- preserve interoperability with MCUs which do not support multicast.

The basic architecture of the proposed solution consists of a thin protocol layer which makes a Reliable Multicast service appear to MCS implementations as a point-to-point service (Figure 2). This approach has the advantage of allowing us to use unaltered MCS implementations.

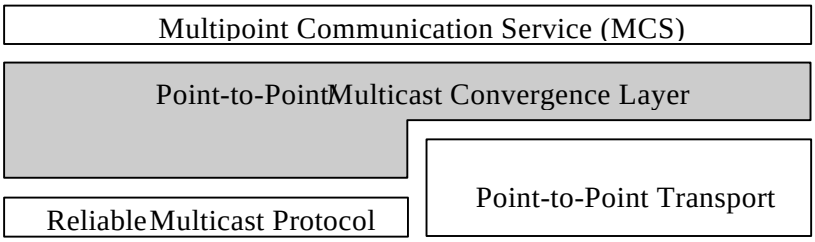

Figure 2: Convergence Layer Interfaces

The Convergence Layer (CL) shown in Figure 2, is responsible for determining whether peer MCUs (i.e., MCUs which share an MCS connection with the local MCU) support multicasting. The layer also maps service primitives from MCS to/from either multicast or point-to-point stacks.

The Convergence Layer we have designed only uses multicast to support regular channel data. All other PDUs are carried over point-to-point connections respecting the domain hierarchy.

When an MCU is a participant in a channel (i.e., some attached users in its sub-tree have subscribed to the channel), it either joins or creates a multicast group for that channel. To maintain data ordering without duplicating MCS's functionality and introducing additional connections, the use of multicast groups is limited to connected parts of the domain tree, called *multicast islands* (as shown in Figure 3 below). Hence a channel can have one or more associated groups in the overall domain tree.

The Convergence Layer Protocol

CL entities analyze MCS PDUs they receive from their local MCS implementation (i.e., received at their service boundary) and PDUs received from peer MCUs. On connections with other CL implementations, CL entities embed MCS PDUs in CL PDUs along with additional CL specific information. In all but one case (see below), embedded MCS PDUs are forwarded unaltered by CL to their local MCS implementation.

To avoid introducing unnecessary overhead, the CL protocol relies to a large extent on a proper implementation of the MCS protocol. CL is also dependent on a reliable multicast transport service, although it is independent of the exact protocol implementing that service.

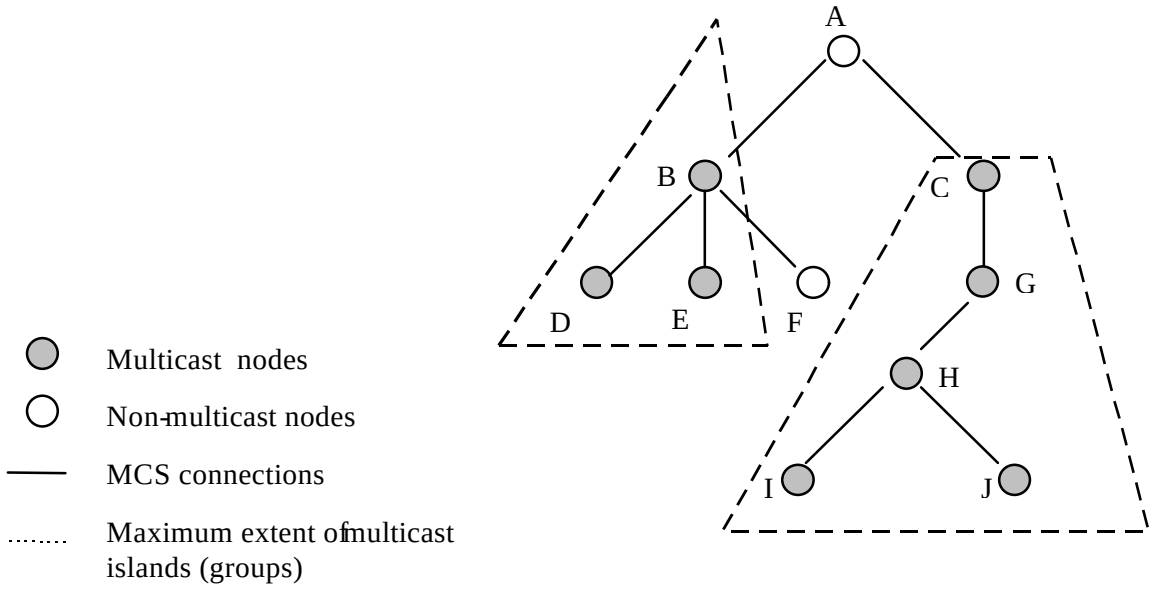

Figure 3: Scope of multicast addresses in an MCS domain

The main functions of the CL protocol are to:

- determine which MCUs support the CL protocol,
- manage the creation/dissolution of multicast groups, and
- prevent duplication of data which stems from the combined use of multicast transport and the multicast function of MCS.

In this section, we describe the basic principles of CL operation. A more detailed description of the CL protocol is given in appendix A.

CL Support Discovery

The convergence layer protocol needs to establish whether peer MCUs are multicast capable or not. This is done when the local MCS implementation attempts to establish the first point-to-point connection with a distant MCU. At that point the CL entity of the originator sends a probe to the recipient on a well known port. If the distant MCU does not answer the probe or if it explicitly rejects it, the CL reverts to point-to-point operation on that domain connection, i.e., it acts strictly as a pass-thru layer. If the peer is multicast capable and accepts multicast operation, the newly established point-to-point connection will be used to carry all control information and encapsulated, non-data MCS PDUs.

Channel/Multicast Group Management

Multicast groups are established when channels are created or joined. The CL layer analyzes MCS PDUs received from the local MCU and from peer MCUs. According to the MCS protocol, the MCU which requests the creation of a channel needs to request from the domain's top MCU a unique ID for the new channel. The corresponding request flows upward to the top provider and is

followed by a downward MCS PDU carrying the ID of the new channel. A multicast group will be created in each multicast island along the confirm path by the root of the island. All MCUs on the path from the root of the island to the MCU which initiated the original request will join the group.

The procedure to join a channel in the MCS protocol is similar. The node which needs to join the channel, sends a request up the tree. This requests travels toward the domain's root up to the first node which is already part of the channel. That node issues a downward confirm. CL implementations along the path join the corresponding group.

A CL entity leaves a multicast group when the local MCS implementation sends its parent a request to leave the associated channel. According to the MCS protocol, this will only happen when no subordinate MCU is still part of the channel.

The allocation of multicast addresses is not fully specified in our design and is somewhat orthogonal to the CL protocol. In this context it is important to reduce the number of collisions, i.e., situations where two channels share the same group ID, although this problem is handled by parameters of data PDUs sent over multicast groups. Several schemes for reducing/eliminating the possibility of collisions exist: reducing the TTL (time-to-live) parameter on multicast packets, one-to-one mapping of channel Ids (unique) onto multicast addresses, address reservation protocol, etc. To be effective these schemes need to strike a balance between the need to limit collisions and the available multicast address space.

Channel Data Transfer

The convergence layer protocol needs to analyze PDUs sent/received by/from the local MCS implementation in order to determine how to map channel data. This data may have been originated by local users or may be relayed downward or upward by the local MCU. Since data is carried over multicast groups, relayed data which has already been seen on the multicast group needs to be suppressed in order to prevent data duplication.

Detection of duplicate data is primarily based on the fact that MCS channel data PDUs carry the unique identifier of the data originator. The core of the convergence layer protocol is to maintain enough information about the channel/group mapping and MCS user locations to identify data requests which need to be suppressed. CL implementations keep track of user locations based on the MCS PDUs they receive.

All data received on a multicast group is handed to the local MCS implementation. The basic rules for processing data requests issued by the local MCS implementation are as follows:

I. requests addressed to a peer accessible only through a point-to-point connection are forwarded to that peer;

II. a subordinate MCU which is multicast capable is arbitrarily designated as a *trigger;* data requests to multicast capable peers other than the trigger MCU are always suppressed by the CL;

III. when the CL receives data on a multicast group (i.e., from peer MCUs), it can determine that the user who originally issued the data is reachable through the multicast group and is not local; the received data is handed to the local MCS implementation, but all data requests indicating that the original source of the message is one of those users are suppressed;

IV. other data requests are sent out on the multicast group associated with the channel.

Rule I above allows data to cross multicast islands and to reach non-multicast capable MCUs. Rule II insures that there will be no duplication of data on origination when the local MCS implementation attempts to dispatch it to all of its subordinate MCUs. Rule III suppresses all relaying of data inside the same multicast island.

Implementation Specifics

The Multimedia Protocols Group implemented the Convergence Layer in the following environment:

- Transport Control Protocol (TCP) for point to point communications (for non-multicast conference MCUs and transfer of control information for all participants);
- Reliable Multicast Protocol (RMP) for data transmission to "multicast island" member MCUs;
- The layer and its support classes are implemented as C++ objects, with a reliance on Rogue Wave Corp.'s Tools.h class library for basic storage classes and complex types (i.e. String);
- Windows 95 operating system;

Appendix A: Convergence Layer Protocol - Procedures

This appendix is a high level description of the service primitives, PDUs, and operations of the CL protocol. The reader is assumed to be very familiar with the MCS protocol.

CL Upper Interface Primitives

CL service primitives are essentially those defined in T.123, i.e., the primitives generally used to describe a point-to-point, connection-oriented transport service.

Primitives sent from MCS to CL:	*Primitives sent from CL to MCS*
T-Connect_req	T-Connect-ind
T-Connect_rsp	T-Connect-conf
T-Data_req	T-Data-ind
T-Disconnect_req	T-Disconnect-ind

CL Lower Interface Primitives with Point-to-point Transports

Since CL acts as a transparent convergence layer when multicast is not available, it uses the same primitives with the underlying point-to-point transport as those in T.123 (see above).

CL Lower Interface Primitives with Reliable Multicast Transports

CL relies on the service typically expected from reliable multicast transport protocols.

Primitives sent from CL to Multicast Transport:	*Primitives sent from multicast Transport to CL*
MCT-CreateGroup-req	MCT-Data-ind
MCT-JoinGroup-req	MCT-AbortGroup-ind
MCT-SendData-req	

CL PDUs

CL uses the following PDUs. All CL PDUs except CL-MC-Data-RI are embedded in a T-Data-req sent over the point-to-point connections CL maintains with multicast capable peers. CL-MC-Data-RI is only used for channel data on multicast groups. To simplify notations, this mapping will not be explicit in the rest of this appendix.

PDUs	*Description*
CL-Probe-RI	*Probe request/indication - used to determine whether peer implements CL.*

CL-Probe-RC	*Probe confirm - used to confirm support of multicast*
CL-Connect-RI	*Connection request/indication - used to notify peer of additional pseudo-connection (in MCS terms).*
CL-Connect-RC	*Positive confirm of Connect-RI.*
CL-Disconnect-RI	*Disconnect request-indication.*
CL-Data-RI	*Contains MCS data PDU sent over control connection - used for non-multicast MCSPDUs.*
CL-LGrp-RI	*Left group request/indication.*
CL-AGrp-RI	*Available group request-indication - used to communicate channel/group mappings.*
CL-MC-Data-RI	*Multicast channel data request/indication.*

CL Procedures for Upper Layer Requests

T-Connect-req

When MCS issues a T-Connect-req, the convergence layer will issue a probe to the peer. If this operation fails, it will use exclusively point-to-point operation for this and all future connections with the peer. Otherwise, it will establish a point-to-point connection with the peer and use it for control information and to carry non-data MCSPDUs, and will use multicast groups to carry channel data.

```
if exist point-to-point connection to peer
      send T-Connect- req to peer
else if first T- Connect_req for this MCS connection
      send CL-Probe-RI.
      start connection-timer - later treat timer expiration
      as a negative CL-Probe-RC.
else
      send CL-Connect-RI to peer
```

T-Connect-rsp

This will translate either into a CL-Connect-RC (for connections with multicast capable peers) or into a T-Connect-resp otherwise.

```
if peer is  multicast capable
      send CL-Connect-R C
else
      send T-Connect- rsp
```

T-Disconnect-req

According to the MCS protocol, a node may maintain up to four connections with a peer node. The effect of this primitive is to close one of these connections. If the peer implements CL, all four connections are mapped onto the same control connection, plus multicast groups representing channels. CL-Connect-RI and CL-Connect-RC PDUs are used to maintain a usage count; only when that count becomes zero is the control connection actually broken.

```
if peer is  multicast capable
      send CL-Disconnect-RI
      if last pseudo-connection
            send T-Disconnect-  req on the control connection.
else
      send T-Disconnect-  req
```

T-Data-req

When MCS issues a T-Data-req(mcs-pdu) request to CL, if the peer is not CL capable, the primitive is forwarded to the point-to-point control stack, unchanged. Otherwise, the following operations will be performed by the convergence layer.

```
case mcs-pdu = SDin or SDrq      //  channel data
      if control connection is oldest connection maintained
         by this node and we have never received data from
         the source of the   SDin/ SDrq
            send CL-MC-Data-  req( mcs_pdu)
      else
            do nothing (i.e., suppress request)


case mcs-pdu = CCcf/CJcf/CAin   // channel join confirm
      if channel creation failed
            send CL-Data-RI(  mcs-pdu)
      else
            if group already exists for channel id
                  issue  MCT-JoinGroup-req to MC stack
            else
                  issue  MCT-CreateGroup-req to MC stack

            send CL- AGrp( mcs-pdu) to peer


case mcs-pdu = CLrq              // channel leave request
      send CL- LGrp( mcs-pdu) to peer
      issue a  MCT-LeaveGroup-req to the MC stack.


case mcs-pdu = AUcf/MCcf         // Attach user/Domain Merge
      add the user(s) to the set of subordinate users

      send CL-Data-RI(  mcs-pdu) to peer


case mcs-pdu = DUin              // Detach user request
      remove the user from the set of users from which we
      have received  multicast data in the past

      remove user from the set of subordinate users

      send CL-Data-RI(  mcs-pdu) to peer
```

CL Procedures for CL PDUs

CL PDUs are received on control connections from peer MCUs.

CL-Probe-RI

When a CL implementation receives a CL-Probe-RI, it has the option to accept/reject usage of CL on the related MCS connection. The actual connect indication will be issued to the local MCS on receipt of the CL-Connect-RI.

```
if acceptable connection
      send CL-Probe-RC(OK) to peer
else
      send CL-Probe-RC(not OK) to peer
```

CL-Probe-RC

If the probe confirm from the peer is positive, CL will issue a CL-Connect-RI to the peer, otherwise it will revert to point-to-point operation.

```
if CL-Probe-RC(OK)
      send CL-Connect-RI to peer
else
      revert to point-to-point operation on this
      connection (i.e. act as a pass-  thru layer).
```

CL-Connect-RI

Issue a connect indication to the local MCS. Increase the usage count of the control connection.

```
      issue T-Connect- ind to local MCS  impl.
      increase usage count of control connection.
```

CL-Connect-RC

Issue a connect confirm to the local MCS implementation.

```
      issue T-Connect- conf to local MCS  impl.
```

CL-Disconnect-RI

Issue a disconnect indication to the local MCS implementation. Decrease the usage count of the control connection.

```
      issue T-Disconnect- ind to local MCS  impl.
      adjust usage count of control connection.
```

CL-Data-RI

Issue a data indication to the local MCS. The contents of the CL PDU is a MCS PDU which is not handled by CL.

```
issue T-Data- ind to local MCS  impl.
```

CL-AGrp-RI

The local CL joins the group defined in the channel/group association if it is not already part of that group. The MCS PDU embedded in the CL-AGrp-RI is forwarded to MCS.

```
if local MCS not part of the group,
      issue a  MCT-Join-Grp-req to the MC stack.

issue a T-Data- ind to the local MCS
```

CL-LGrp-RI

The local CL leaves the group defined in the channel/group association.

```
if this is the island root and there are no more
   multicast subordinates on this channel
      issue a  MCT-LeaveGroup-req to the MC stack.
issue a T-Data- ind to the local MCS.
```

CL-MC-Data-RI

This is data received on a multicast group. Since the same embedded MCS data PDU (i.e., SDrq/SDin) is sent to all members of the group, some members need to change it into an SDrq, others into an SDin, depending on the respective position of the original source of the message and the receiver.

```
if  CL-MC-Data-RI( SDin) and source is a subordinate user
      issue a T-Data- ind( SDrq) to the local MCS   impl.
else if CL-MC-Data-RI(  SDrq) and source is not a
                  subordinate user
      issue a T-Data- ind( SDin) to the local MCS   impl.
else
      issue a T-Data- ind to the local MCS   impl.

add source user  to set of users from which we have
received  multicast data.
```